Compliant Task Execution and Learning for Safe Mixed-Initiative Human-Robot Operations

Shuonan Dong*, Patrick R. Conrad†, Julie A. Shah‡, Brian C. Williams§
Massachusetts Institute of Technology, Cambridge, MA, 02139, USA
David S. Mittman¶, Michel D. Ingham‖, and Vandana Verma**
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109, USA

We introduce a novel task execution capability that enhances the ability of in-situ crew members to function independently from Earth by enabling safe and efficient interaction with automated systems. This task execution capability provides the ability to (1) map goal-directed commands from humans into safe, compliant, automated actions, (2) quickly and safely respond to human commands and actions during task execution, and (3) specify complex motions through teaching by demonstration. Our results are applicable to future surface robotic systems, and we have demonstrated these capabilities on JPL's All-Terrain Hex-Limbed Extra-Terrestrial Explorer (ATHLETE) robot.

I. Introduction

Our goal is to provide a more natural, commonsense way of interacting with the next generation of NASA robotic systems. One example of such a robot is the All-Terrain Hex-Limbed Extra-Terrestrial Explorer (ATHLETE), a vehicle developed at JPL to support robotic and human missions on the surface of the Moon. Previously, robots have been simple enough to be controlled through teleoperation or through scripts composed of many low-level motion commands. However, more complex robots, like ATHLETE or the Vecna Battlefield Extraction-Assist Robot (BEAR) shown in Figure 1, have too many degrees of freedom for direct teleoperation or commanding to be practical. For example, a script for the ATHLETE rover, a 36-degree of freedom robot, is composed of many unintuitive low-level commands such as

> "move joints on limb 1 with speed .8 in absolute coordinates: hipyaw 0.510 hippitch 0.221 kneepitch -1.605 kneeroll 0.000 anklepitch -0.189 ankleroll -1.960."

To enable more efficient and flexible interaction, we developed a different control paradigm that incorporates verbal commands, shared written instructions, and demonstration by example. These modes of communication occur naturally to humans during collaborative tasks. This control paradigm allows the user to specify intuitive, goal-directed commands, and then teach the robot their meanings through demonstrations. An intelligent executive then interprets the commands in the context of shared written instructions, and transforms them into low-level motion commands to the robot.

We developed two related task executives called Chaski and Drake, and demonstrated their performance on the ATHLETE platform. Both executives allow temporal reasoning with choices that the robot is allowed to make: Chaski focuses on multi-agent collaboration and Drake focuses on allowing arbitrary choices between sub-components of a plan. Additionally, we also developed a motion learning capability that enables a robot to autonomously perform motions learned from human demonstrations in new environments.

*PhD Candidate, Department of Aeronautics and Astronautics, 32-224, AIAA student member.
†PhD Candidate, Department of Aeronautics and Astronautics, 37-438.
‡Assistant Professor, Department of Aeronautics and Astronautics, 77 Massachusetts Ave, Cambridge, MA 02139.
§Professor, Department of Aeronautics and Astronautics, 32-227.
¶Senior Member of Technical Staff, Planning Software Systems, M/S 301-250D.
‖Technical Group Supervisor, System Architectures and Behaviors Group, M/S 301-490, AIAA senior member.
**Technologist, Advanced Robotic Controls, M/S 321-151.

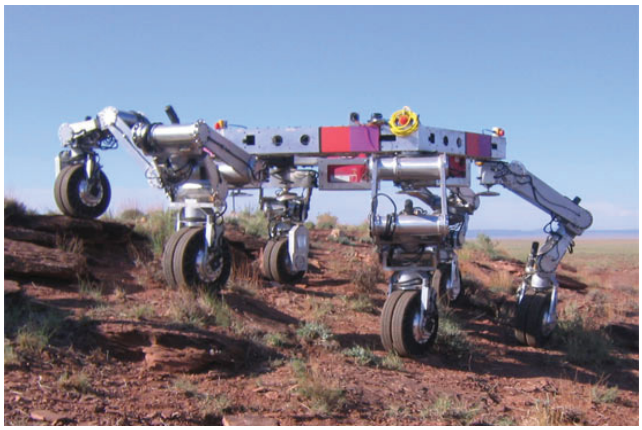

Figure 1. ATHLETE rover (left), Vecna BEAR (right)

The remainder of this paper will present our problem description, describe the context of prior art, discuss each component of our approach, and present the results from our hardware demonstrations with ATHLETE.

A. Problem Setup

Our goal is to enable reliable and safe operations of both humans and robots performing tasks in-situ, and provide natural and efficient interaction through tele-operation. To this end, we present the following subproblems:

1. Map a human's goal-directed commands, such as "Robot: help me dig a 5m long trench on the west side of the habitat," into flexible plans composed of low-level actions conditioned on safe environment states.

2. Safely adapt to human-initiated, goal-directed changes in the plan, such as "Robot: instead help me dig the trench on the east side of the habitat," or "Robot: instead dig a 7m trench." To be useful to the astronaut, the robot must quickly and safely choose actions to respond to these changes in operator-specified goals, while considering possible conflicts with the human's concurrent tasks and changes in the environment.

3. Learn low-level complex spatial and temporal actions through demonstration and tele-operation, thereby reducing time and effort required to execute a mission. For example, an ATHLETE operator would specify an action through a tele-operation interface. ATHLETE then learns this action as a primitive, safe and compliant action that may be executed automatically in the future.

The problem can be set up with the following interfaces:

- Offline inputs: (1) an intuitive high-level specification of the behavior of the robot encoded in Reactive Model-based Programming Language (RMPL), and (2) human demonstration of motion-level activities for complex continuous motions not already known to the robot.

- Online inputs: (1) real-time human coaching commands from human (voice or teleoperated), (2) real-time feedback from ATHLETE's sensors (joint angles, camera data), and (3) status updates among agents.

- Output: a dynamic execution policy that guarantees compliant, temporally consistent, logically valid actions given real-time feedback from the environment

B. Relation to Prior Art

The first subproblem is mapping goal-directed commands from humans into safe, compliant, automated actions. The goal of the compliant execution capability is to promote reliable and safe operations in which humans and robots perform tasks in close physical proximity. In developing this capability, we build on recent work on compliant task execution and flexible plans.[7] Our approach extends flexible plan execution techniques, previously used only for discrete systems, to continuous systems, such as legs and arms, which operate safely around humans. We will develop our compliant plan execution capability by building on prior work in execution of temporally flexible plans,[10,12] and our recent work in this area.[7,13–15]

The second subproblem is enabling quick and safe response to human commands and observations during task execution. Our approach extends prior work in dynamic plan execution[2,10,12,15] to allow fast, safe responses to changes in the environment. We generate families of safe and successful plan executions and switch between executions at run-time to flexibly respond to unexpected events. Plan execution in an uncertain environment is made tractable by generating policies focused on reacting quickly to unsafe situations. For example in the ATHLETE trenching and transport scenario, our capability will represent contingencies such as the astronaut entering the workspace, and enable ATHLETE to quickly respond by halting trenching and switching to another activity such as transporting the habitat. This approach will allow ATHLETE to make decisions that lead to the safe execution of its goals, and with certain models of uncertainty, can guarantee success in the face of unpredictable events.

The third subproblem is learning parameterized spatial task-plans through tele-operated demonstration. Our approach is based on recent work in motion learning,[4] and encodes a motion using a representation called a probabilistic flow tube. In the past, constraint-based flow tubes have been used in the context of planning and execution with continuous motions, to represent sets of trajectories with common characteristics.[6,8] We also introduce the ability to automatically determine relevant variables of a motion directly from observed training sequences, so that the motions relationship to these variables can be preserved when autonomously executing the motion in a new situation. Our problem is similar to that of task space selection,[11] although instead of the human-robot correspondence problem, we are more interested in detecting what relevant features, if any, exist in the environment, and when they become relevant in a motion.

II. Approach

Our system architecture is shown in Figure 2. A user, caricatured as an astronaut, provides written instructions about the tasks in the Reactive Model-based Programming Language (RMPL),[18] which gets stored as a library of plans. Complex continuous activities in the plans that need further description is obtained through activity learning, in which a set of user's teleoperated robot motion sequences is paired with a user-provided activity label to generate a probabilistic representation of the motion stored in the probabilistic activity library. During execution, the flexible executive can execute a plan from the plan library, whose lower level activity control to the robot is provided in the activity library. While execution is proceeding, the user can interact with the executive by issuing goal-directed voice commands.

A. Task Executives

Allowing the user to specify intuitive, goal-directed commands to the robot requires an intelligent executive to interpret the commands and transform them into low-level machine instructions. The process begins with the user giving directions to the robot in the Reactive Model-based Programming Language (RMPL).[18] This language allows high-level instructions at a level closer to where engineers practically think, such as "move the box," in a convenient Java-like syntax. Additionally, the user can provide constraints that the system needs to obey without giving specific instructions on how to do so, allowing engineers to think at a high level about what they need. Here, we mostly deal with temporal constraints: restrictions on the duration of activities and completion deadlines. The executive is then responsible for reasoning through the constraints set forward in the plan and executing the activities correctly.

The executive performs constraint reasoning to determine all the implications of the user's requests and then adjusts the execution in real-time in response to the outcomes of events, making it more robust to disturbances. The executive is given explicit direction about what the user requires, allowing it to determine the total set of possible plans and schedules that meet those requirements. Then, the executive is empowered to shift between different executions rather than being bound to a single type of reasoning or a single plan

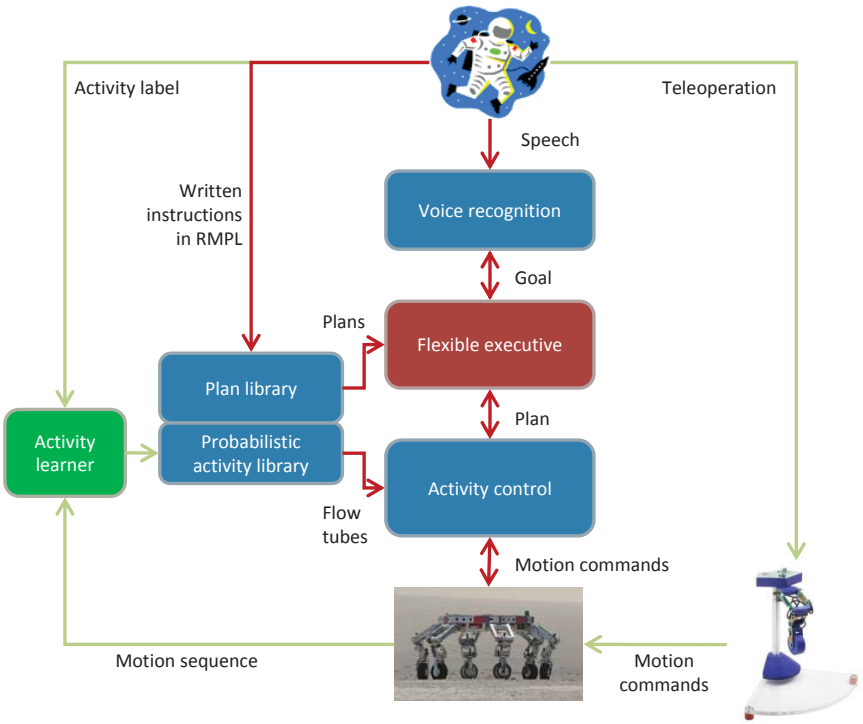

Figure 2. System architecture

provided before even beginning. A reactive executive, which can adjust the execution sequence on the fly as a plan unfolds, can guarantee the correct execution of a plan even with some disturbances. To make this possible, the executive is broken into a compiler and a dispatcher. The input plan in RMPL is converted into a temporal constraint formalism to allow reasoning on it, and the compiler performs a type of reasoning on it, reformulating it into a dispatchable form. The dispatcher works with the dispatchable form to efficiently schedule the start and end of the activities in the task, adjusting the plan as it observes the start and end of events.

Specifically, our executives called Drake[2] and Chaski,[16] are designed to execute Temporal Plan Networks (TPN), which RMPL programs can encode. Temporal Plan Networks specify a set of activities, choices, and temporal constraints among activities. From this plan, the executive extracts a Disjunctive Temporal Network (DTN), a temporal reasoning framework that includes choices, which might include choices between task assignments or possible goals.[3] The compiler then evaluates the possible options, reducing the plan to a *dispatchable form*, defined as a form where only one-step propagations are required at run-time. Then, the executives dynamically execute the plan, adapting to disturbances if possible or immediately notifying the system if a failure is observed. Drake and Chaski improve on the state of the art by performing this reasoning with compact data structures. Drake is designed to handle general types of choices in plans, and Chaski is specialized for multi-agent plans with resources.

B. Learning from Demonstration

Generating low-level trajectories to achieve complex continuous motions, such as "unwind the cable," can be a challenge. We reduce time and effort required to do this by teaching actions through demonstration. For ATHLETE, we use an interface device called Tele-robotic ATHLETE Controller for Kinematics (TRACK),[9] shown at the lower right of Figure 2, to teleoperate the robot through each desired activity multiple times.

Based on the demonstrations, our learning algorithm generalizes from the examples a set of motions that will likely achieve the desired activity. This is compactly encoded in a representation called a probabilistic flow tube,[4] as illustrated in Figure 3. Flow tubes have been used in the context of planning with continuous motions to represent sets of trajectories with common characteristics.[6,8] We have incorporated this concept to motion learning by developing the probabilistic version, represented by a mean trajectory and correspond-

ing covariances. Geometrically, the breadth of a probabilistic flow tube represents flexibility in the robots desired movement, enabling it to optimize additional performance criteria or recover from disturbances. We choose the probabilistic flow tube representation because it models motions close to how humans do, since they are directly based on human-generated trajectories. For robots designed to work in the field with other humans nearby, such as in the case of ATHLETE, it is important that any autonomous behavior should be executed in a way humans expect, and not in a way that could cause alarm for the human, even if it means executing a less optimal behavior.

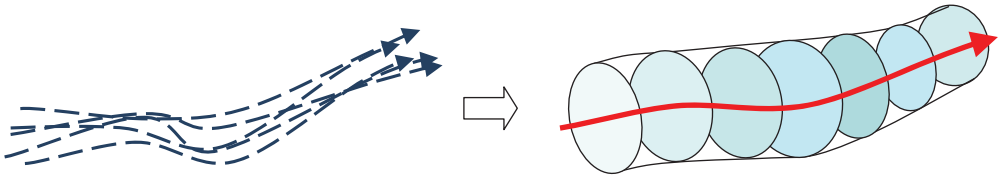

Figure 3. The probabilistic flow tube on the right is a learned generalization of the demonstrated motions on the left.

Another key feature of our learning system is the ability to autonomously determine what parameters, if any, are relevant in a demonstrated motion. We use the term 'parameter' to describe qualitative features of a motion. For example, a relative motion to move an object two feet to the right has two relevant parameters: the initial location of the object, and that the relative motion distance is two feet. Parameter identification uses clustering to determine if patterns exist across different training samples at object contact points. Our current system considers the following types of motions: (1) absolute motions, such as "move to position (5,5)," (2) motions relative to an object, such as "move to the box," (3) motions relative to the robot, such as "move two feet to the right," (4) or unparameterized motions, such as "wave arm." Without any prior knowledge about what type of motion a demonstration might be, the system is able to determine which relevant parameters, if any, exist in the motion by observing patterns across training demonstrations. We assume the locations of the objects and robot in the environment can be determined through sensing, such as GPS or vision processing.

During demonstrated training, the initial positions of objects in the environment are not fixed or predefined, but rather sensed when needed. When the robot is prompted to execute the learned motion in a new situation, any object positions identified as relevant parameters are sensed, and the system generates a probabilistic flow tube using training trajectories with initial conditions similar to those observed. This flow tube is scale- and rotation-normalized to be used in the new situation.

III. Experimental Setup and Scenarios

Our demonstration scenarios involved manipulation tasks performed by two limbs on the ATHLETE robot. We provided online voice commanding though the CMU Sphinx Toolkit.[1] Task descriptions were written in RMPL, and teleoperation was performed through a Tele-robotic ATHLETE Controller for Kinematics (TRACK).[9] We teleoperated the robot through the desired complex activities several times.

We demonstrated the performance of our system through two task scenarios, both involving limbs 1 and 6 of the ATHLETE robot. In the first task scenario described in Figure 4, a human and ATHLETE cooperate to collect rocks. ATHLETE is equiped with interfaces on the limbs of its wheels to attach tools such as a gripper. The first step in the cooperation requires the human to attach a gripper to one of ATHLETE's limbs. ATHLETE then moves the gripper to a rock of interest. Because the rock is small and sensing is limited, the human fine tunes ATHLETE's gripper location through voice commanding. Next, the human attaches a bin to ATHLETE's other available limb. ATHLETE can then pick up the rock while moving the bin to a position to receive the rock. Parallel activities like these two have no ordering constraints, so they can be executed in any order or simultaneously. Finally, ATHLETE deposits the rock into the bin, and stores the bin for transport.

Our second scenario involves a construction task where the robot chooses different activities based on the current situation. The entire task, presented in Figure 5, needs to be completed within 5 minutes. ATHLETE begins by picking up a box, with fine-tuning help from the operator, and placing it on top of a platform. Then, if the robot has enough power left, it should perform an inspection with the other camera-

1. Human: attach gripper to ATHLETE's left limb
2. ATHLETE: prepare to pick up rock with gripper
3. Human: fine tune ATHLETE gripper location
4. Human: attach bin to ATHLETE's right limb
5. In parallel:
    1. ATHLETE: move bin to receive rock with right limb
    2. ATHLETE: pick up rock with left limb
6. ATHLETE: deposit rock in bin
7. ATHLETE: store bin for transport

Figure 4. Deposit rock into bin task description

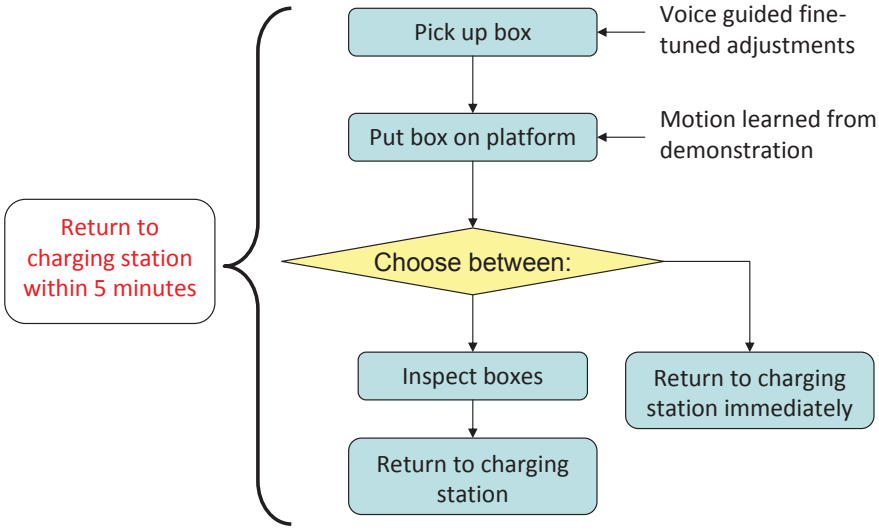

Figure 5. Move box on top of platform task description

```
method run (){
   [0m,5m] sequence {
      limb1 pick up box
      limb1 put box on platform
      choose {
         limb6 inspect box
         noOperation
      };
      return to charging station
   }
}
```

Figure 6. Move box on top of platform task specification in RMPL

equiped limb, and return to its charging station. If it does not have enough time, it should return to its charging station immediately. Figure 6 shows the corresponding RMPL task specification. We can see that it is written at the same high level as the task flow chart.

IV. Results from demonstrations

We used the Chasi executive to perform the collaborative rock collection task described in Figure 4. Chaski successfully demonstrated tight temporal synchronization of the two ATHLETE limbs, modeled as independent agents, under uncertainty in how long various actions would take. Specifically, the two limbs had to be synchronized to pass a rock held by one limb into a sample container held by the second, which requires that both limbs act at the same time. Figure 7 shows the resulting execution of the collaborative task.

Figure 7. Deposit rock to bin demonstration. Notice the human operator collaborates by giving the robot the gripper in frame 3. The operator also attaches the bin in frame 11.

We used the Drake executive to perform the construction task described in Figure 5. This task has an optional inspection activity that could be performed if sufficient time remained after completing the construction. The executive could not predict in advance if there would be sufficient time for the extra task, requiring that it wait until run time to make the decision online. We tried several demonstrations of the construction task. Each time, the operator takes different amount of time to issue voice commands to fine tune the first activity of picking up the box. If the human took too long, the executive successfully chose the option to return to charging station immediately, rather than perform the inspection activity.

In each of the two scenarios, there were continuous activities for which specifying the constraints would be more complex than directly teaching the motion to the robot. In the rock collection task, we taught the robot the motion to move a rock to the bin, and in the construction task, we taught the robot the motion to move the box on top of the platform. All demonstrations were made using the TRACK interface device. For each demonstrated activity, we showed the robot the motion 5 times, each time with different intial positions of the objects involved. During execution, the objects were in previously unseen locations. The autonomous execution of the learned "move rock to bin" motion can be seen in frames 10-13 of Figure 7. The five demonstrated trajectories of the "move box to platform" motion can be seen in Figure 8, and the autonomous execution of the motion can be seen in Figure 9.

V. Conclusion

We demonstrated two task executives, Chaski and Drake, on the ATHLETE platform, with the ability to teach the robot complex continuous motions. Currently, intelligent task executives are not widely adopted in space applications. While the tasks we performed during the tests were short and mostly serve as proofs of concepts, demonstrating on advanced hardware platforms like ATHLETE raises the technology readiness level of our capabilities, and aids in making the technology feasible. Furthermore, it lends weight to previous simulated experiments indicating the usefulness and performance of the algorithms behind the executives.

The ability for an operator to teach a robot desired motions can allow for more intuitive interactions. Our demonstration of motion learning on ATHLETE shows the potential feasibility and benefits of this mode of

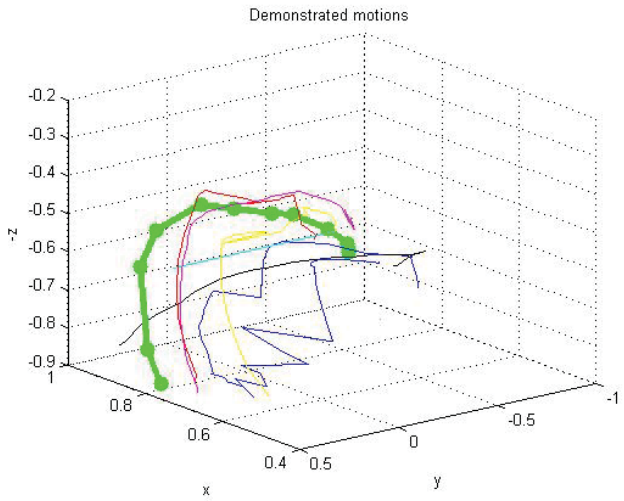

Figure 8. "Move box to platform" ATHLETE limb 1 end-effector motion. Thin lines are demonstrated motions. Thick line is the trajectory of the learned motion applied to a new situation. Movement starts from the lower left and ends toward the upper right. Note the different initial locations.

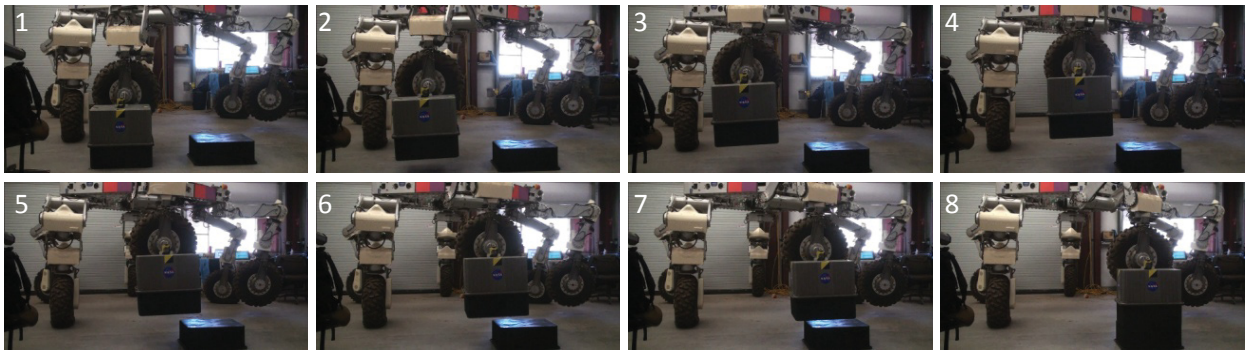

Figure 9. Video capture of autonomously executed motion

operation in future human-robot missions. In our demonstrations, we showed that only a few demonstrations may be necessary to obtain desirable generalized motions that can be autonomously executed in a wide range of new situations.

Acknowledgments

This research was carried out at MIT, funded by the JPL Strategic University Research Partnership program, and at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. This work was partially supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program.

References

[1]Bawab, Z. A., "An Analysis-by-Synthesis Approach to Vocal Tract Modeling for Robust Speech Recognition," Ph.D. Thesis, ECE Department, CMU, September, 2009.

[2]Conrad, P., Shah, J., and Williams, B. C., "Flexible Execution of Plans with Choice," International Conference on Automated Planning and Scheduling, Thessaloniki, Greece. Sept. 2009.

[3]Dechter, R., Meiri, I., and Pearl, J., "Temporal constraint networks," Artificial Intelligence, 49:61-95, 1991.

[4]Dong, S. and Williams, B. C., "Motion Learning in Variable Environments Using Probabilistic Flow Tubes," Proceedings of International Conference of Robotics and Automation, Shanghai, China, 2011.

[5]Dvorak, D., Michel D. Ingham, Morris, J.R., and Gersh, J. "Goal-based operations: An overview," In Proc.of the Infotech@Aerospace Conf. and Exhibit, May 710, 2007. 9.2.3

[6]Hofmann, A., Williams, B. "Robust execution of temporally flexible plans for bipedal walking devices," In Proc. ICAPS-06.

[7]Hofmann, A. G. and Williams, B. C., "Exploiting Spatial and Temporal Flexiblity for Plan Execution of Hybrid, Under-Actuated Systems," Proceedings of the 21st National Conference on Artficial Intelligence, Boston, MA, July 2006, pp. 948-955.

[8]Li, H. and Williams, B. C., "Generative Planning for Hybrid Systems based on Flow Tubes," Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), Sydney, Australia, 2008.

[9]Mittman, D. S., Norris, J. S., Powell, M. W., Torres, R. J., McQuin, C., Vona, M. A., "Lessons Learned from All-Terrain Hex-Limbed Extra-Terrestrial Explorer Robot Field Test Operations at Moses Lake Sand Dunes, Washington," Proceedings of AIAA SPACE, 2008.

[10]Morris, P., Muscettola, N. "Execution of temporal plans with uncertainty," In Proc. AAAI-00.

[11]Muhlig, M., Gienger, M., Steil, J. J., and Goerick, C., "Automatic Selection of Task Spaces for Imitation Learning," Proceedings of IROS, 2009.

[12]Muscettola, N., Nayak, P., Pell, B., Williams, B. "To boldly go where no AI system has gone before," AI 103(1):5-48. 1998.

[13]Stedl, J. "Managing Temporal Uncertainty Under Limited Communication: A Formal Model of Tight and Loose Team Communication," S.M. Thesis, MIT, 2004.

[14]Shah, J., Stedl, J., Williams, B.C., Roberston, P. "A Fast Incremental Algorithm for Maintaining Dispatchability of Partially Controllable Plans," In Proc. ICAPS-07.

[15]Shah, J., Williams, B.C. "Fast Dynamic Scheduling of Disjunctive Temporal Constraint Networks through Incremental Compilation," In Proc. ICAPS-08.

[16]Shah, J., Conrad, P., and Williams, B. C., "Fast Distributed Multi-agent Plan Execution with Dynamic Task Assignment and Scheduling," International Conference on Automated Planning and Scheduling, Thessaloniki, Greece. Sept. 2009.

[17]Tsamardinos, I., Muscettola, N., Morris, P. "Fast transformation of temporal plans for efficient execution," In Proc. AAAI-98.

[18]Williams, B. C., Ingham, M., Chung, S. H., and Elliott, P. H., "Model-based Programming of Intelligent Embedded Systems and Robotic Space Explorers," Invited Paper, Proceedings of the IEEE: Special Issue on Modeling and Design of Embedded Software, vol. 91, no. 1, pp. 212-237, January 2003.